REMARKS

Claims 1-28 are pending, of which Claims 1, 13, 25, 26 and 28 are independent. All claims have been rejected under 35 U.S.C. § 103(a) based on U.S. Patent No. 5,732,271 to Berry in view of Hostetter et al., "Curl: A Gentle Slop Language for the web," World Wide We Journal, Spring, 1997. These rejections are respectfully traversed. Reconsideration is requested.

Rejections under 35 U.S.C. § 103(a)

In order to establish a prima facie case of obviousness under 35 U.S.C. § 103(a), three criteria must be met. There must be: (1) some suggestion or motivation to modify the reference or to combine reference teachings, (2) a reasonable expectation of success, and (3) a teaching or suggestion of all the limitations of the claim. See M.P.E.P. § 2143. For the reasons discussed below, it is respectfully submitted that the Examiner has not established a prima facie case of obviousness for any of the present claims, and that therefore, the present claims are allowable.

Preferred embodiments of the invention relate to a technique that allows type processing of option values during compilation without having to store full option values in memory. A class is defined that supports an option data structure. An instance of the class has references to option values without preallocating memory space for full option values. The option data structure includes type descriptions of the option values. The option data structure can, for example, be a linked list. During compilation, the compiler uses the type descriptions to process operations on the option values. In this way, the invention can enable compile-time processing of options with type information but without preallocated storage.

By way of contrast, Berry is silent as to how memory is allocated. It appears that Berry is consistent with traditional object-oriented techniques, as Berry does not suggest modifying them in any way. Therefore, in Berry, memory space is presumably allocated for each value, similar to object-oriented languages, regardless of whether the value has been set on the object. As a result, memory is allocated for all the potential properties of an object, even if the properties are never used by the object.

More particularly, in conventional object-oriented systems such as Berry, a variable or field name is associated with (1) a declared, compile-time type, and (2) preallocated storage, which holds a value (possibly a pointer to another data structure) of that type. By way of contrast, the present invention eliminates the need for item (2). This allows for compile-time processing of options with type information but without preallocated storage.

At page 3 of the Office Action, the Examiner states that in Berry only attribute values contained in the derived object need memory preallocation, while values that are set in the prototypical object do not require preallocation in the derived object. It is respectfully submitted, however, that in Berry, each attribute value that can be set on an object has a preallocated field in that object that will be set to the attribute value if a value is assigned for that attribute on that object. Even when this field is not set, it still occupies its own dedicated memory space in the object. It is true that some types of attribute values (such as variable-length character strings) may require additional allocation when they are set, but even in those cases, the object upon which the attribute value can be set must have a preallocated field (typically 4 bytes) to contain the address of the additional storage allocated for the attribute value. Although Berry discusses using fields (e.g. field "fBackground" in Berry's coding example at col. 5, ll. 1-12), Berry does not discuss any technique for avoiding the allocation of this basic field. Even if the amount of preallocated storage per object is only 4 bytes for each attribute value that could be set on the object, storage space considerations can become significant when an object has a large number of potential attribute values that could be set on the object but are not set.

For example, when designing a graphical user interface, it can be desirable to have a vast number of style and behavior properties that can apply to groups of objects in the display. In a system of the Berry type, storage space would be allocated for each of the potential properties. Each of the potential property values would be allocated storage space during the life of the object, and if none of those values were actually set on the object, the storage may be considered wasted.

The present invention, however, enables instances of a class to include references to option values without preallocation of memory space for the full option values. Storage space is allocated when the option value is actually set on the object. In this way, the invention addresses memory issues that are not contemplated by Berry. Accordingly, Berry does not relate to the claimed technique for defining a class which supports an option data structure having, in instances of a class, references to option values without preallocation of memory space for the full option values, as set forth in Claims 1, 13, 25, 26 and 28.

The Examiner correctly notes that Berry does not discuss the claimed compile-time technique, which uses type descriptions in the option data structure to process an operation on an option value. The Examiner cites Hostetter to show this feature. Although Hostetter provides for the definition of variables and named fields with type declarations that are used by a compiler in processing and translating programs, Hostetter does not provide type information for compile-time processing of option values.

Previous systems that supported option-like mechanisms, provided options in library packages. These systems, however, processed the option data types at run-time.

Conversely, the present invention provides options that work like fields but offers a different space/time tradeoff. Specifically, the present invention defines a class which supports an option data structure that enables instances of the class to have references to option values without preallocation of memory space for the full option values. The option data structure includes type descriptions of option values, which are used by the compiler to process operations on the option values. These features are not taught by either Berry or Hostetter. Thus, neither Berry nor Hostetter, taken separately or in combination, discuss the claimed technique that allows type processing of option values during compilation without having to store full option values in memory.

As such, it is respectfully submitted that the Examiner has not made a prima facie case under 35 U.S.C. § 103(a) because the combination of Berry and Hostetter does not teach every aspect of the claimed invention, namely:

- defining a class which supports an option data structure having, in instances of the class, references to option values without preallocation of memory space for the full option values, the option data structure including a type description of the option values, as set forth in Claims 1, 13, 25, 26 and 28 respectively; and
- during compilation, using the type description in the option data structure to process an operation on the option value, as set forth in Claims 1, 13, 25, 26 and 28 respectively;
- option data structure comprises a linked list of option items having option values, as set forth in Claims 7, 19 and 27 respectively; and
- type description is used to check the declared type of a value to be set in a set operation, as set forth in Claims 11 and 24 respectively.

Therefore, it is respectfully requested that the rejection of Claims 1, 13, 25, 26 and 28, and their respective dependent claims, under 35 U.S.C. § 103(a) be withdrawn.

Regarding Objections to the Specification

The Examiner objected to the written description because [incr Tk], described at page 1, lines 14-15, is not referenced properly. The Applicants appreciate the Examiner's attentiveness to the accuracy of the written description. In response to the Examiner's objection, the discussion of [incr Tk] in the Specification at page 1, lines 14 through 19, has been amended by the present amendment. In [incr Tk], there is preallocation of memory space for option values and the application has been amended to correct any reference to the contrary. Thus, the Specification has been amended to correct a drafting error. Acceptance is respectfully requested.

The Examiner also objected to the Abstract because it exceeded the 150 word limitation. The Abstract is amended to meet that limitation. No new matter is introduced. Acceptance is respectfully requested.

CONCLUSION

In view of the above amendments and remarks, it is believed that all claims are in condition for allowance, and it is respectfully requested that the application be passed to issue. If the Examiner feels that a telephone conference would expedite prosecution of this case, the Examiner is invited to call the undersigned.

Respectfully submitted,

HAMILTON, BROOK, SMITH & REYNOLDS, P.C.

By _____

James M. Smith
Registration No. 28,043
Telephone: (978) 341-0036
Facsimile: (978) 341-0136

Concord, MA 01742-9133
Dated: 4/26/4